A Simple Case Study of a Grid Performance System

Ruth Aydt, Dan Gunter, Darcy Quesnel, Warren Smith, Valerie Taylor
aydt@uiuc.edu, dkgunter@lbl.gov, quesnel@mcs.anl.gov, wwsmith@nas.nasa.gov, taylor@ece.nwu.edu

Grid Forum Performance Working Group
February 22, 2001

1 Introduction

This document presents a simple case study of a Grid performance system based on the Grid Monitoring Architecture (GMA) being developed by the Grid Forum Performance Working Group. It describes how the various system components would interact for a very basic monitoring scenario, and is intended to introduce people to the terminology and concepts presented in greater detail in other Working Group documents.

We believe that by focusing on the simple case first, working group members can familiarize themselves with terminology and concepts, and productively join in the ongoing discussions of the group. In addition, prototype implementations of this basic scenario can be built to explore the feasibility of the proposed architecture and to expose possible shortcomings. Once the simple case is understood and agreed upon, complexities can be added incrementally as warranted by cases not addressed in the most basic implementation described here.

Some open issues and complex requirements that came up during the discussions are noted, but no attempt is made to address them in this document. In the simple case presented here, it is assumed that all components have the necessary authorization to interact in the manner described. Clearly, authentication and authorization are two very important considerations in a real implementation of any Grid performance system.

2 Scenario

Ten workstations (*ws1 – ws10)* are used as desktop systems by local users and are also available as compute engines for Grid applications. A monitor is running on each of these workstations to measure the CPU load every 30 seconds. The CPU load measurements are all forwarded to a central server (*srvr)* machine on the same local area network as the workstations. A process on the server makes the load information available to systems not located on the local network.

One of the system administrators for the workstations telecommutes from her home in another state – her home machine is named *adminsys.* She continuously graphs the loads of the workstations in a window on *adminsys* to ensure that there are no problems with the *ws* machines.

Further, all of the load measurements are being archived by an archiving service on the machine *archivsys.* The archival data is used by another program to analyze daily system load patterns and to identify time periods when the workstations are heavily utilized so that backups will not be scheduled during those times.

3 Terminology

In this section we define some of the basic terms used by the Grid Forum Working Group and relate them to the simple case study presented in this document.

3.1 Event, Event Type, and Event Data

An *event*[*] is a structure containing one or more items of data that relate to one or more resources. Every event has an associated e*vent type* that uniquely identifies the structure for that particular event. The term *event data* refers to one or more of the items of data making up an event.

In the scenario described above, the machines *adminsys* and *archivsys* are interested in events of type CPU_LOAD. They want to receive CPU_LOAD event data describing the load for systems *ws1* through *ws10.* Depending on the implementation, a single event may or may not contain information for all of the systems. In the implementation outlined below, an event contains the CPU load information for a single system.

3.2 Event Schema

An *event schema* describes the structure for a particular event.

In the basic scenario described in this document, a schema will be defined for the CPU_LOAD event type.

3.3 Producer

A *producer* is a component that makes event data available to other components that are part of the Grid Monitoring Architecture. The producer speaks a standard protocol and generates event data in a standard format. It is possible that there will be multiple standard protocols and formats defined within the GMA. The producer may not be the ultimate source of the data – that source may or may not speak the same protocol and use the same event data format. The GMA is not concerned with defining the protocol(s) and format(s) used by the original sources of the performance data.

In our scenario, a process on *srvr* is a producer and makes event data available to other components in the Grid performance system being described. Monitoring processes on *ws1* through *ws10* are the sources of the measurement data, but they are not producers.

3.4 Consumer

A *consumer* is a component in the Grid Monitoring Architecture that receives event data from a producer. It speaks a standard protocol and expects the event data to be in a standard format.

In the basic scenario described, processes on *adminsys* and *archivsys* are consumers of the event data produced by *srvr.* The *adminsys* consumer process will graph the per-host CPU_LOAD measurements. The *archivsys* consumer process will write the event data to disk for later examination.

[*] *Events,* as defined and used in this document, are implicitly *performance events.* We make no attempt to define or discuss other types of events.

3.5 Directory Service

A *directory service* is a searchable component in the Grid Monitoring Architecture used to store and forward information that is of general interest to other components in the system. The directory service can be queried through a variety of search mechanisms and returns information matching the specified selection criteria. The directory service may in practice be implemented as a set of distributed, interconnected individual directory services under the control of different organizations. We anticipate using the Lightweight Directory Access Protocol (LDAP) to interface with the directory service. See [1,2,3] for information on LDAP.

In the GMA, several distinct types of information will be stored in the directory service and we refer to the directories for each information type by a unique name. The actual implementation may place all entries in a single directory service, but conceptually we believe it is easiest to think of them as independent directories. Here we define only those directories that are necessary to implement the basic scenario.

3.5.1 Event Type Directory

The *Event Type Directory* contains event schema for the various events in the system. The Event Type Directory does *not* contain actual events. For each event type there will be one schema in the Event Type Directory -- within the system all events of the same type must have the same structure.

The Event Type Directory can be searched by event type. It can also be searched by event element name, for example, "return all the event types that contain an element named *cpuload_measurement*".

To support the basic scenario described, the CPU_LOAD schema must be included in the Event Type Directory.

3.5.2 Event Producer Directory

The *Event Producer Directory* contains information about producers and the event types they provide.

All producer information in the Event Producer Dictionary is structured according to an *Event Producer Schema.* In contrast to the Event Type Directory, which contains the event schema but not the actual events, the Event Producer Directory *does* contain the actual producer information records and not just the schema for those records.

Consumers use the Event Producer Directory to locate producers of events they are interested in receiving. There are many possible ways a consumer might want to search for producers in the Event Producer Directory including: by event type, by producer, by host where the measurement originated, or by any number of other keys. The choice of what search keys should be supported is an open question.

For the basic scenario outlined in this document, the Event Producer Directory will contain one or more entries indicating that CPU_LOAD event data for *ws1* through *ws10* is available from a producer on *srvr.*

3.5.3 Event Consumer Directory

The *Event Consumer Directory* contains information about consumers, the event types they accept, and the services they provide.

All consumer information in the Event Consumer Dictionary is structured according to an *Event Consumer Schema.* As with the Event Producer Directory, the Event Consumer Directory *does* contain the actual consumer information records and not just the schema for those records.

Producers use the Event Consumer Directory to locate consumers that provide services of interest. As with the Event Producer Directory, the search keys that should be supported for the Event Consumer Directory remains an open question.

To support the basic scenario described in this document, the archival process on *archivsys* will register with the Event Consumer Directory as a consumer that accepts all event types and provides an archival service.

4 Implementation

In this section we describe, at a fairly high level, the steps necessary to implement the basic scenario on the Grid Monitoring Architecture. Through this description we hope to give the reader a clear idea of how the GMA components cooperate, and to provide a framework from which prototype implementations can be developed to test various protocols and formats.

4.1 Event Schema and Event Type Directory

To implement the basic Grid performance system described, we must first define the event schema for the CPU_LOAD event. This schema will be stored in the Event Type Directory where it can be located and used to interpret data values in CPU_LOAD events. We use a representation-independent format to define the schema here:

Event Type		**Event Description**
CPU_LOAD		CPU load measurement for a single host
Element Name	**Element Data Type**	**Element Description**
cpuload_measurement	double	measured CPU load
hostname	string	host where measurement was taken
timestamp	ASCII timestamp	time measurement was taken
producer	URL	URL for producer generating event data

CPU_LOAD Schema

As defined, a CPU_LOAD event has four data elements that contain the CPU load measurement, the host the measurement relates to, the time the measurement was made, and an identification of the producer that made the event data available.

4.2 Event Producer Directory

The next step in the implementation process is for the producer, *srvr,* to add entries to the Event Producer Directory, advertising that it will provide CPU_LOAD event data for *ws1, ws2, ... ws10.* As stated earlier, we expect to use LDAP to interact with the Event Producer Directory and insert these entries.

We have not yet reached a consensus on the contents of the Event Producer Directory entries, that is, the Event Producer Schema has not yet been set. We believe further discussion and experimentation are required to correctly identify an appropriate Event Producer Schema, and the version presented here should not be interpreted as a standard.

For the purpose of this simple case study we list the type of information that might be included in the Event Producer Directory entries. Two Event Producer Directory entries are shown, those for the CPU load data from *ws1* and *ws2*. Similar entries will exist for *ws3* through *ws10*.

Field Name	Value
Producer	srvr:portXX
EventType	CPU_LOAD
Host	ws1
Parameters	NONE
Filters	NONE
Access	OPEN
ConnectionProtocol	SimpleXML
DataFormat	XML
Producer	srvr:portXX
EventType	CPU_LOAD
Host	ws2
Parameters	NONE
Filters	NONE
Access	OPEN
ConnectionProtocol	SimpleXML
DataFormat	XML

Event Producer Directory Entries

In the simple case study presented in this document, the consumer on *adminsys* is interested in CPU_LOAD data for any of the *ws* machines. To support this scenario, the Event Producer Directory will be searched for entries with an EventType of "CPU_LOAD" and a Host of "ws1" through "ws10". The Producer field specifies where to contact the producer to receive events of interest.

The remaining Event Producer Directory fields are not explicitly used in this simple case study, but are included to show possible extensions that are discussed in other working group documents. Parameters could be used to indicate that the producer would allow the consumer to specify some parameters, such as frequency of event record transmission. The Filters field could be used to indicate that the producer has some built-in filtering capabilities, such as sliding window average computations. The ACCESS field is intended to provide different levels of access to the event data that is being produced – for example, make data available only to consumers within the same organization or make data available to anyone.

The ConnectionProtocol field could be used to specify which of several standard protocols the producer understands, for example SimpleXML or SNMP. The DataFormat field could be used to specify which of several standard data formats the producer is capable of generating, for example XML, ULM, SDDF, SNMP. A consumer may be fluent in a limited set of the possible protocols and formats and consequently would only consider connecting to producers that "speak" those protocols and formats.

4.3 Event Consumer Directory

Another step in the implementation process is for the archiving consumer on *archsys* to advertise its existence. As with the Event Producer Directory entries, the Event Consumer Schema describing the contents of the Event Consumer Directory entries has not yet been finalized.

For the purpose of this simple case study we show the type of information that might be included in the Event Consumer Directory entries. An entry for the archiving consumer on *archsys* is shown.

Field Name	Value
Consumer	archsys:portYYY
EventType	*
Service	archive
Access	Producer=*.mydomain.edu
ConnectionProtocol	SimpleXML
DataFormat	XML

Event Consumer Directory Entry

The Consumer field specifies where to contact the consumer process, the EventType field indicates the types of events the consumer is willing to accept, and the Service field shows the service or services the consumer provides. The Access, ConnetionProtocol, and DataFormat fields have the same meaning as they did in the Event Producer Schema. Note that values containing *'s indicate wildcards.

In our case study, the producer process on *srvr* will search the Event Consumer Directory on startup to find a consumer that will accept and archive the CPU_LOAD events related to machines *ws1* through *ws10.* Assuming *srvr* is in "mydomain.edu", the producer process on *srvr* will be able to contact the consumer process on *archivsys* and request that the consumer subscribe to the CPU_LOAD events for *ws1* through *ws10* that are made available by the producer.

4.4 Consumer/Producer Communication Established

Now that the directory service contains the event type schema, event producer information, and event consumer information, the Grid performance system is ready to share measurement information taken on resources in one part of the Grid with processes running on other systems in the Grid

In particular, for our simple case study the load-graphing tool running on *adminsys* posts a query to the Event Producer Directory requesting any CPU_LOAD events for machines *ws1* through *ws10.* The query returns ten matches, all with the same Producer contact values. Using the connection protocol retrieved from the Event Producer Directory, the load-graphing tool on *adminsys* connects to the producer process at *srvr:portXX*, and subscribes to the CPU_LOAD events for *ws1, ws2, ... ws10.*

After starting up, the producer process on *srvr* queries the Event Consumer Directory to find a consumer that offers archival services for CPU_LOAD events. Assuming *srvr* is in "mydomain.edu", the entry for the archival service on *archivsys* is returned. At this point, the producer process on *srvr* contacts the archival consumer process on *archivsys* and requests that the consumer subscribe to the producer's CPU_LOAD events for *ws1* through *ws10*.

The consumer/producer communication channel is established when a consumer subscribes to a producer. That is, the consumer notifies the producer that they want to receive certain events until further notice. In the simple case study scenario, the subscription from the archival consumer is *producer-initiated*.

4.5 Producer Sends Event Data to Consumers

Once the consumers have subscribed to the events of interest, the producer sends CPU_LOAD event data to the consumers until the subscription is cancelled.

The event data is sent in a standard protocol, which could be either the same as the connection protocol or a different protocol negotiated during the connection process. The event data is sent in the format advertised in the Event Producer Directory. If the producer advertised that it can generate multiple data formats, then the consumer may specify which of those formats to use in the subscription request.

Sample event data encoded in XML is shown here, with white space added for readability:

```
<CPU_LOAD>
   <cpuload_measurement>30.09</cpuload_measurement>
   <hostname>ws1</hostname>
   <timestamp>2001-01-30T20:33:05.003Zp.001a.5</timestamp>
   <producer>http://srvr.mydomain.edu/producerXX</producer>
</CPU_LOAD>

<CPU_LOAD>
   <cpuload_measurement>22.98</cpuload_measurement>
   <hostname>ws9</hostname>
   <timestamp>2001-01-30T20:34:15.07Zp.01a.5</timestamp>
   <producer>http://srvr.mydomain.edu/producerXX</producer>
</CPU_LOAD>
```

The load-graphing tool receives the event data and updates the display for each host with the appropriate measurements. The archiving service receives the event data and writes it to the archive for later analysis by the backup-scheduling program.

5 Summary

We have described a very basic performance monitoring scenario in a Grid environment, defined terms used within the Grid Forum Performance Working Group and related those to the scenario, and outlined at a fairly high level how the scenario could be implemented with the components defined in the Grid Monitoring Architecture. This basic scenario ignores may important and complex issues that are critical to a fully functional Grid Performance System in the interest of presenting basic concepts and providing a starting point for discussion and prototype implementation experiments.

6 Open Issues

In this section we note many of the issues that were raised and set aside when this simple case study was developed, recognizing the need to return to them in future discussions and documents. These are loosely arranged based on the system components they relate to, but in many cases the issues cross boundaries between components and therefore the categories should not be considered to be exact.

6.1 Event Schema and Event Type Directory

- No units or accuracies are specified for any of the event elements.
- We may also want to include other Event Identifiers, in addition to the Event Type. For example, and OID.

- In the simple case study, all event elements are required. In practice, there may be event types for which some elements are optional.
- If an event element has units associated with it, how would two events that are identical except for the units associated with one of the elements be implemented? A particular example of this would be if the CPU_LOAD event element "cpuload_measurement" had units associated with it and in some cases the units were "percent busy over last minute" while the others they were "percent busy over the last five minutes". Some options:
    - As two distinct event types, perhaps CPU_Load_sec and CPU_Load_fivesec. Here the units are implicit in the event type. No extra event data is sent from the consumer to the producer to indicate units, and no conversions need to take place when the data is received. The downside is the possible proliferation of event types with only minor differences between them.
    - As a single event type with an additional element specifying the measurement units:
      Event Element < name=measurement_units, type=integer, description=number of seconds in measurement window>. This method would require passing the additional "measurement_units" information with each event, increasing the amount of data transmitted. The 'dual-duty' makes interpreting the measurement value more complicated on the consumer side, for example in labeling the displays.
    - As a single event type, with the units somehow specified for all event records from a particular hostname/producer combination. This could be done if the producer and consumer event data formats supported the sending of partial events. In the basic case study the event data contained all data items making up an event. If instead only some data items were sent, the consumer would be responsible for "assembling" the full event from several partial events. This implementation choice would not require larger events, but would add complexity to accommodate the partial events and the interpretation of the event data in the context of the appropriate units on the consumer end.
    - As a single event type, with registration of the units in the Event Producer Directory when the producer registers that it will generate CPU_LOAD events for a given host. This implementation choice would not require larger events, but would add complexity to accommodate the additional information in the Event Producer Directory and the interpretation of the event data in the context of the appropriate units on the consumer end.

6.2 Event Producer Directory

- Unsure how consumers will want to locate information within the event producer directory, that is what keys to search on and implications on directory service organization. Some possibilities are:
    - …/Site/Producer/Host/<EventType>
    - …/Site/Host/Producer/<EventType>
    - …/Site/EventType/Host/Producer/…

  or some combination of these…
- Support for Parameters in the directory, in producers, and consumer requests, needs to be fully designed and described. Some Parameters may be associated with the Event Type, (for example, remote host to contact in a ping request), while others may vary depending on what a given producer will support.
- Support for Filters in the directory, in producers, and consumer requests, needs to be fully designed and described. The Corba Event Service discussion on filters that support the use of logical expressions based on event element and event parameter values should be reviewed for possible adoption or implementation ideas.
- ConnectionProtocol and DataFormat specification in the directory, and in producer/consumer interactions needs to be fully designed and described.
- Access specification in the directory is simply defined as OPEN. This needs to be fleshed out and perhaps integrated as part of the directory service implementation hierarchy. An alternate suggestion was

an enumeration of Access types like OPEN; RESTRICTED; GROUP; etc. that would be put into the tuple as shown. This needs more debate. Someone also mentioned "aci attributes" in LDAP that address access issues and those should be investigated.
- Need to come to agreement on the Event Producer Schema

6.3 Event Consumer Directory

Most (all?) of the open issues for the Event Producer Directory have counterparts for the Event Consumer Directory.

6.4 Other Directory Services

- Additional directories will likely be needed. For example, a *Type Directory* to define recognized data types and a *Unit Directory* to define recognized units.

6.5 Producer/Consumer Exchanges

- What data formats to support
- What wire protocols to support
- In the simple case study the consumer subscribed to certain event data – the GMA also supports a one-time query interface
- Security issues – how to verify that the consumer is who they claim to be and that they are authorized to receive the requested data. When to do this verification.
- Considerations regarding data volume. See discussion in section 6.1 on partial events as one possible way to reduce event data volume. Also, some discussions about binary and compressed formats to reduce volume.
- Interpretation of event data, sometimes called "context". In general, measurement data is interpreted relative to a larger set of information than what is typically transmitted in the event data itself. This may include things like operating system version, clock accuracy, frequency of event data transmission, etc. How is this critical information conveyed, archived, used?

References

[1] T. Howes, M. Smith, G. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Technical Publishing, 1999.
[2] T. Howes and M. Smith. *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing, 1997.
[3] Innosoft's *LDAP World* webpage with links to various on-line LDAP documents: http://www.innosoft.com/ldapworld